IMPORTANTE:

- **Crear un proyecto con su Apellido y nombre.**
- **Añadir comentarios identificando a qué ejercicio e inciso corresponde cada función. No alcanza con comentar en el Main o en los prototipados, EN CADA FUNCIÓN DEBE COMENTARSE A QUÉ EJERCICIO E INCISO CORRESPONDE. *NO SE CORREGIRÁ CODIGO NO IDENTIFICADO.***

Una importante empresa de alquiler de disfraces nos ha encomendado realizar un pequeño sistema para manejar el stock de disfraces en sus distintas sucursales.
La información se encuentra almacenada en el archivo subido al campus, el cual hay que descargar y copiar dentro de la carpeta del proyecto del examen. Se trata de un archivo de estructuras "stRegistroDisfraz" que responde a la siguiente estructura de datos:

```
typedef struct {
    int idSucursal;
    char nombreSucursal[25];
    char nombreDisfraz[25];
    char generoDisfraz[25];
    int stockDisfraz;
}stRegistroDisfraz;
```

Se nos pide desarrollar la siguiente funcionalidad (**a los fines de facilitar la corrección, desarrollar todas las funciones en el archivo main.c en lugar de hacerlo en librerías**):

1. Codificar las estructuras y funciones que hagan falta para **recorrer el ARCHIVO y pasar los datos a una LISTA (doblemente enlazada) de LISTAS (simplemente enlazadas) (LDL),** clasificando los datos de la siguiente manera:

En cada nodo de la **lista principal** se ubica c/u de las sucursales de la empresa, de acuerdo a la siguiente struct:
 	
`typedef struct{`
`  int idSucursal;`
`  char nombreSucursal[25];`
`}stSucursal;`
	En cada nodo de la llista ppal. también se encuentra una **sublista** con todos los disfraces que tiene en stock cada sucursal, de acuerdo a la siguiente struct:
	
	`typedef struct{`
	`  char nombreDisfraz[25];`
	`  char generoDisfraz[25];`
	`  int stockDisfraz;`
	`}stDisfraz;`

   Para realizar esto, se deberá hacer lo siguiente, **modularizando de acuerdo a la responsabilidad de cada TDA**:
   - **a)** Codificar las estructuras necesarias para manejar la lista principal y las sublistas.
   - **b)** Recorrer el archivo, y por cada dato leído, deberán copiarse los campos correspondientes para crear las variables de tipo stSucursal y stDisfraz.
   - **c)** Buscar la sucursal a la cual corresponde el disfraz a cargar en la LDL.
   - **d)** Agregar una nueva sucursal si ésta no existe.
   - **e)** Agregar un nuevo disfraz a la sucursal que corresponda. **Los datos deben ser insertados en la forma en que resulte más EFICIENTE hacerlo** (al principio?, en orden?, al final?).

   35 puntos

2. **Mostrar** todos los elementos de la **estructura compuesta**, permitiendo visualizar cada sucursal con sus disfraces en stock.
   - **Las sucursales (la lista principal) deberán ser mostradas en órden inverso** (**desde la última hacia la primera), y los disfraces desde el primero hacia el último.**
   - **Modularizar de acuerdo a la responsabilidad de cada TDA**.
   - A los fines de facilitar la tarea, se entrega el código correspondiente al mostrado de la sucursal:

```
puts ("\n************************************ SUCURSAL ************************************\n");
printf ("\nId de la Sucursal……..: %d \n", dato.idSucursal);
printf("\nNombre de la Sucursal: ……:%s \n", dato.nombreSucursal);
puts ("\n**********************************************************************************************\n");
```

   - y del disfraz:

```
puts ("\n----------------------------------------------------------------------------------------\n");
printf("\nNombre del disfraz.....: %s \n ", dato. nombreDisfraz);
printf("\Genero al que pertenece.....: %s \n ", dato. generoDisfraz);
printf("\nStock del disfraz.....: %d \n", dato. stockDisfraz);
puts ("\n----------------------------------------------------------------------------------------\n");
```

10 puntos

3. Codificar las estructuras y funciones que hagan falta para **recorrer la LDL y cargar los datos en un ARBOL.**
   - **Modularizar de acuerdo a la responsabilidad de cada TDA**:
   - Cada nodo del árbol utilizará una **struct de tipo stRegistroDisfraz** (es decir: contendrá tanto los datos del disfraz como de la sucursal a la cual pertenece).
   - En el árbol los disfraces deberán **insertarse por orden alfabético de acuerdo al campo "nombreDisfraz")** (ojo: puede haber disfraces con nombres repetidos; ej: Shrek puede estar repetido porque hay modelos para adultos y modelos para niños)**.**

20 puntos

4. **Mostrar** el **ARBOL.**
   - Los DISFRACES deberán mostrarse EN ORDEN ALFABETICO (pensar qué modo de mostrado usar).
   - **Modularizar respetando la responsabilidad de cada TDA.**

8 puntos

5. Nuestro cliente desea poder **buscar determinado disfraz de acuerdo a dos parámetros*: el nombre del disfraz y su género,* y saber si lo posee o no.**
   - Ojo! recordar que el árbol está ordenado de acuerdo al campo "nombreDisfraz", y que ahora hay que buscar por 2 campos: "nombreDisfraz" y "generoDisfraz". Y recordar también qué criterio fue utilizado en el ejercicio 3 para cargar los disfraces con nombres repetidos (de qué lado del árbol fueron cargados)
   - **Pensar bien cómo buscar para no perder la eficiencia que tienen las BÚSQUEDAS BINARIAS en los Arboles Binarios de Búsqueda**.

17 puntos

6. Hacer una función **main**()
   - Para hacer esto, cree las variables que considere necesarias e invoque las funciones (de forma directa o indirecta) como corresponde en cada caso.
   - Muestre los resultados cada vez que sea necesario.

10 puntos